De l'activité collaborative aux micro-outils : illustration avec PLACID, une plate-forme agent

Alain-Jérôme Fougères, Jean-Pierre Micaëlli

De l'activité collaborative aux micro-outils : l'exemple de PLACID, une plate-forme agent

Alain-Jérôme Fougères [1] et Jean-Pierre Micaëlli [2]

[1] Laboratoire M3M, Université de Technologie de Belfort-Montbéliard, http://m3m.utbm.fr
alain-jerome.fougeres@utbm.fr
[2] Laboratoire RÉCITS, Université de Technologie de Belfort-Montbéliard
jean-pierre.micaelli@utbm.fr

Résumé

Cet article entend présenter le concept de micro-outils développé pour assister l'activité de concepteurs de systèmes mécaniques, ainsi que la plate-forme logicielle PLACID qui supporte ces micro-outils. Un micro-outil est destiné à une tâche très précise, dont il va faciliter ou améliorer l'exécution, tout en laissant à son utilisateur toute latitude dans l'organisation de son activité à une échelle plus globale. PLACID est une plate-forme qui repose sur le paradigme agent. Celui-ci concerne aussi bien la modélisation et le développement des différentes couches de la plate-forme que celles de ses interfaces. À ces particularités de la plate-forme s'ajoutent des contraintes fortes de souplesse et d'adaptabilité, visant à faciliter l'intégration de nouveaux outils d'assistance à l'activité collaborative (réalisation de tâches et résolution collective de problèmes). Pour illustrer notre approche nous présenterons la conception de micro-outils intégrables dans un atelier coopératif d'analyse fonctionnelle.

Mots clés : micro-outil, agent logiciel, conception collaborative, plate-forme collaborative.

1 Introduction

Les activités collaboratives prennent une place croissante dans le monde industriel ou tertiaire. L'une des causes de ce phénomène récent tient sans doute à l'élévation de la complexité des conditions d'offre des produits et des services. Une automobile requiert de plus en plus de conception, donc de services incorporés. Le moindre dossier bancaire suppose la mobilisation de nombreux professionnels du *front office* (relation avec la clientèle) ou du *back office* (traitement des dossiers). Assister ces deux activités suppose de mettre à disposition des professionnels de ces secteurs de véritables bureaux virtuels qui nécessitent une palette d'outils permettant :

- la communication interpersonnelle ou de groupe (de types synchrones et/ou asynchrones) ;
- l'activité de groupe, sa structuration spatio-temporelle et la coordination des tâches réalisées par les uns et les autres. Il s'agit alors de gérer dans l'interaction la distance spatiale entre les acteurs et leur distance temporelle, notamment la séquentialité et/ou le parallélisme de la réalisation des tâches ;
- la distribution et le partage d'informations, d'applications, de ressources le plus souvent nombreuses, variées et hétérogènes.

Différents spécialistes interviennent dans le développement d'organisations, de compétences, d'outils, etc., à visée collaborative. Leur association avec des informaticiens répond à la volonté de proposer des concepts, des méthodes et des outils susceptibles de répondre aux fonctions listées ci-dessus.

Dans cette perspective, nous proposons la conception de micro-outils (µ-outils) destinés aux activités de conception collaborative (ou co-conception). Le concept de µ-outil [27] correspond à des applications logicielles légères, faciles d'utilisation, insérables dans un environnement partagé et connectables entre elles. Une plate-forme agent (PLACID : ***P**late-forme **L**ogiciel d'**A**ide à la **C**onception **I**nnovante et **D**istribuée*) a été développée pour supporter l'usage coopératif de µ-outils. La plate-forme offre un contexte technique susceptible d'apporter une assistance au travail de co-conception, guidé ou non par des processus complexes prédéfinis.

Les µ-outils supportés par la plate-forme ne sont pas nécessairement intégrés dans un processus prédéfini de conception. Leur utilisation peut être simplement ponctuelle, apportant un service bien ciblé dans une phase de conception. Il participe alors pleinement au processus d'émergence intrinsèque à la conception distribuée [13]. Quoiqu'il en soit, chaque µ-outil est connecté au système multi-agent de la plate-forme PLACID par agentification. Ce qui permet de réaliser une interface de communication entre les µ-outils et le système d'information coopératif.

L'objet du présent article est de présenter le cadre d'intégration des µ-outils sur la plate-forme PLACID et les résultats de recherche disponibles à ce jour. Pour ce faire, nous dresserons en section 2 un bref panorama des différents concepts du travail coopératif assisté par

ordinateur (TCAO) mis en œuvre dans les collecticiels, puis nous présenterons le concept original de micro-outil logiciel et le processus de développement que nous avons défini pour celui-ci. Les sections 3 et 4 se focaliseront sur la description de la plate-forme agent PLACID. Nous présenterons son architecture agent, la distribution et la coopération assurée par cette plate-forme, ainsi que l'intégration de µ-outils coopératifs. Enfin, dans la section 5, pour illustrer les potentialités des µ-outils et de PLACID en co-conception, nous décrirons la conception d'un atelier d'analyse fonctionnelle collaborative.

2 De l'activité collaborative aux µ-outils

2.1 Le collecticiel pour base

Le développement des technologies informatiques, l'utilisation des nouvelles ressources sur Internet, ont donné naissance à une palette d'outils dédiés à l'assistance des activités collaboratives. Nous faisons bien entendu allusion au CSCW (*Computer Supported Cooperative Work*) [23], avec notamment le développement des collecticiels.

Pour faire bref, le collecticiel est un logiciel dont la fonction principale est d'assister un groupe d'acteurs (vision activité) ou d'utilisateurs (vision outil) pour qu'il puisse réaliser au mieux son activité menée dans le cadre d'espaces partagés explicites. Cette activité est variée, puisqu'elle concerne aussi bien la conception de produits industriels, que l'enseignement, les relations commerciales, l'évaluation de performance, les jeux, etc. Dans tous les cas d'application, les membres du groupe constitué collaborent à distance, soit au même moment (activité synchrone), soit à des moments différents (activité asynchrone). Il revient aux systèmes informatiques de permettre aux utilisateurs de réaliser leurs tâches collaboratives à partir de postes de travail respectifs ou, ce qui est plus lourd, depuis des installations spécialisées (salles virtuelles de co-conception, par exemple).

Pour s'en tenir à l'état de l'art, les concepts à considérer pour développer un collecticiel sont les suivants [2], [17] :

- **l'inscription spatio-temporelle** (localisation) de l'activité ; un collecticiel réunissant plusieurs acteurs (utilisateurs) distants géographiquement ou non, et/ou travaillant en même temps ou non ;

- **la richesse des modes de coopération**. Ces acteurs (utilisateurs) peuvent coopérer selon des modalités variées. Du point de vue informatique, importe plus particulièrement la coopération asynchrone, en session, en réunion ou étroite. Il y a ainsi *coopération asynchrone* lorsqu'ils interagissent en échangeant des données et en travaillant quand ils sont disponibles (mode de travail autonome). *Coopération en session,* lorsqu'ils travaillent en même temps sur des données qui leur sont propres, tout en restant mutuellement accessibles pour communiquer (l'objectif étant de faire diminuer les délais d'interaction entre les différents acteurs d'un projet). *Coopération en réunion*, lorsqu'ils travaillent et communiquent en co-temporalité, tout en partageant les objets de leurs travaux et discussions (des rôles leur sont attribués, et chacun participe à son tour). *Coopération étroite* lorsqu'ils communiquent et interagissent en temps réel sur tous les objets partagés créés et utilisés au cours de l'activité ou, ce qui est synonyme du point de vue informatique, dans le projet (accroîssement de la coproduction) ;

- **la flexibilité multi-domaines**. Assister l'activité collaborative suppose de développer des outils malléables, adaptés à de multiples configurations ou paramétrables (flexibilité statique et dynamique), que ce soit dans la distribution de données, dans le partage, dans le contrôle d'accès, dans la représentation de l'information, dans la planification pour l'exécution des tâches.

Tout collecticiel doit non seulement répondre aux exigences énoncées ci-dessus, mais aussi assurer les fonctions suivantes [15] :

- pour ce qui concerne la structuration du groupe : favoriser l'implication individuelle dans le groupe, entretenir sa cohésion et favoriser son développement ;

- pour ce qui concerne la conduite de l'activité collaborative au sein du groupe constitué : faciliter le partage de ressources, assister la coordination et la coopération et améliorer la communication.

La figure suivante (Fig. 1) précise cette énumération fonctionnelle en schématisant les liens entre les fonctionnalités de base du collecticiel, à savoir : la collaboration (visio/videoconférence, outils de localisation, d'annotations, de réunion électronique, de décision de groupe, édition conjointe), la mémoire de groupe (base d'informations partagées dans un groupe, bases de documents, etc.), la circulation de documents et la gestion de processus d'activité. Elle reprend, puis complète le modèle *3C* (communication, coopération, coordination) [8], utile pour définir les espaces nécessaires aux artefacts (objets, outils, etc.) supports de la collaboration.

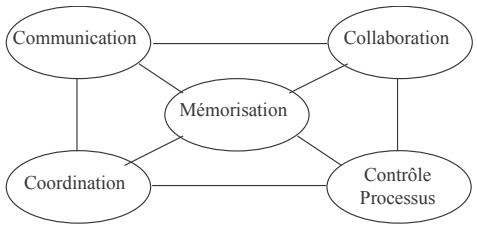

Figure 1. Fonctions du collecticiel complétant le modèle 3C

Le spectre de la coopération que nous considérerons dans la suite de l'article s'étend sur trois niveaux : 1) la coopération naturelle entre acteurs d'une conception, 2) la coopération semi-naturelle entre un acteur de conception et un agent artificiel permettant une interaction avec un artefact, et 3) la coopération artificielle entre agents logiciels intégrés dans un même artefact.

Les rapports entre un groupe de concepteurs et l'organisation d'une activité de conception sont souvent envisagés selon 2 angles complémentaires :

- 2 formes d'organisation sont directement observées : l'organisation planifiée (programmes des concepteurs, intégration des savoirs dans une mémoire centrale, respect de règles initiales) et l'organisation flexible (fonctionnement grâce à la négociation entre des savoirs et des entités hétérogènes) [18].
- 2 modes de conception collective sont observés : la co-conception ou développement conjoint de solution et synchronisation cognitive par construction d'un contexte partagé, et la conception distribuée ou coopération par accomplissement de tâches bien déterminées et synchronisation opératoire (processus de coordination) [4].

2.2 Le concept de µ-outil

Les travaux et les outils relatifs au collecticiel donnent un cadre global d'assistance à l'activité collaborative. Celui-ci s'avère nécessaire mais non suffisant. Il doit être complété par une vision plus « microscopique » [19], focalisée sur les processus opératoires et les nombreuses tâches ponctuelles, itératives, de court terme, menées de façon opportuniste, sur laquelle repose l'activité collaborative. Pour les assister, un concept informatique particulier doit être développé. Pour ce qui nous concerne, nous l'appelons micro-outil [27] (µ-outil), solution alternative et/ou complémentaire aux macro-outils usuels.

Au moins idéalement (Fig. 2), le µ-outil est :

- facile à apprendre et à utiliser (quelques minutes) ;
- peu complexe (architecturalement simple) ;
- évolutif, rapidement implémentable et modifiable, y compris par les utilisateurs eux-mêmes ;
- autonome, mais aussi réactif lorsqu'il est défini pour des processus coopératifs.

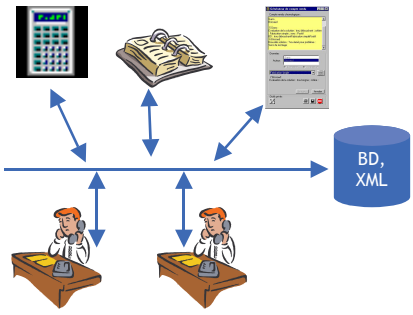

Figure 2. Le concept de µ-outil

Nous considérons deux types de µ-outils : les µ-outils adaptés à la conception de produits et les µ-outils d'aide à la coopération dans les activités de conception ; nous parlerons alors de micro-outils de coopération (MOC). Ces derniers sont distribués entre des acteurs qui interviennent par exemple, dans le cadre d'une co-conception ou d'une évaluation collaborative, selon leurs domaines de compétences (métiers, vues, etc.). Le résultat (produit) de ces actes collaboratifs consiste en un objet dit collaboratif. Il peut s'agir, en conception, d'un objet intermédiaire, d'un cahier des charges, d'un schéma ou d'une feuille de calcul [26]. Nous avons évoqué que le TCAO en tant qu'activité sociale implique la communication, l'organisation (et donc la coordination) et la distribution d'information (échange et partage de connaissances) ; les MOC constitue une réponse à cette nécessité en se focalisant sur le niveau de la tâche.

L'originalité du µ-outil ne réside pas seulement dans sa nature, mais aussi dans son processus de développement. Celui-ci repose en effet sur deux principes :

- **la focalisation**. La Théorie de l'activité (TA) [29], [16], fondement de notre ddémarche d'analyse de l'activité [19], définit trois niveaux d'activités, de tâches et d'opérations. Ce qui permet d'établir les trois co-relations suivantes : les collecticiels correspondent au niveau des activités, les µ-outils au niveau des tâches et les fonctionnalités à celui des opérations (Tableau 1). Ainsi les µ-outils permettent de décomposer un logiciel en un ensemble de modules adaptés à la réalisation de tâches élémentaires. Plusieurs de ces µ-outils peuvent être associés pour accomplir des tâches plus complexes ;

Niveau	Cible de la TA	Volition	SIC associé
Macro	Activité	Motivation	Collecticiel
Micro	**Action/Tâche**	**But**	**µ-outil**
Nano	Opération	Conditions	Fonctionnalité

Tableau 1. Niveaux de l'activité & systèmes d'information coopératifs (SIC).

- **la classification**. Tout µ-outil peut être décrit selon la démarche méthodologique : *identification – classification – qualification – quantification.*

Les deux principes précédents permettent de dresser un paradigme de la nature et du développement du µ-outil :

- son usage est individuel ou collectif (cas des activités collaboratives ou distribuées). Les tâches qu'il assiste sont supposées pouvoir être agencées au sein d'un plan d'actions. Du point de vue du développement logiciel, cela signifie qu'il est nécessaire de bien spécifier les conditions d'usage du µ-outil, en décrivant notamment le cycle de vie des objets traités et leurs conditions de partage ;
- son interaction avec l'acteur (utilisateur) porte principalement sur l'acquisition de données (objets de l'activité), leur mise en relation, avec des moyens graphiques par exemple, puis leurs accès et leur gestion ;
- son identification et sa description sont le fruit d'un travail participatif (utilisateur, développeur) et pluridisciplinaire ; la réalisation de maquettes étant recommandée comme facilitateur d'échanges d'idées ;
- son développement respecte les principes d'une démarche de qualité logicielle.

En suivant l'ensemble de ces principes, nous pouvons dresser un premier plan du processus de développement du

µ-outil (Fig. 3). Celui-ci débute par l'analyse de l'activité et aboutit au produit logiciel correspondant, accompagné de sept documents normalisés qui constituent la mémoire, la « trace détaillée » [19], de leur projet de conception.

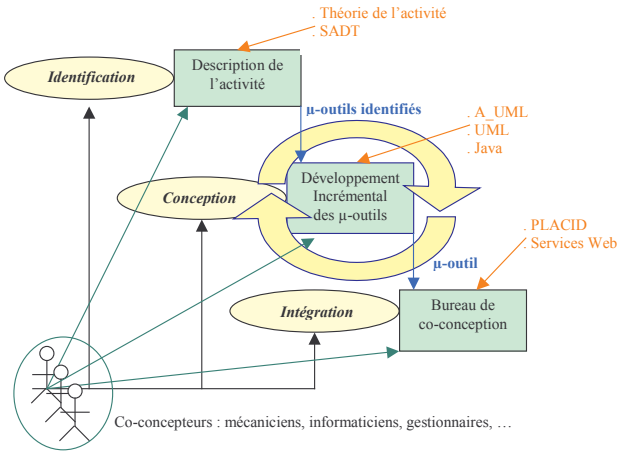

Figure 3. Processus ICI de développement du µ-outil : exemple d'un µ-outil dédié à la co-conception

Pour être encore plus précis, trois grandes phases structurent ce processus coopératif de développement nommé *ICI* (*Identification, Conception, Intégration*) :

- la phase d'*identification* renvoie aux niveaux supérieurs de l'ingénierie système. Elle nécessite une collaboration étendue de tous les acteurs du développement (développement participatif). Elle consiste à identifier, parmi les tâches qui composent une activité spécifique celles, qui pourront être instrumentées, puis à les spécifier ;
- la phase de *conception* vise, selon une approche incrémentale, propice à l'établissement d'un dialogue permanent entre les différentes disciplines participant au développement, à élaborer l'architecture du µ-outil et de ses composants, à développer ces composants et à les tester (UML/Java), puis à valider l'interface utilisateur [12] ;
- la phase d'*intégration* vise, dans notre cas particulier, à interfacer le µ-outil avec PLACID. Cette plate-forme agent est reliée à un ORB (*Object Request Broker*) chargé de la gestion des échanges et du partage des informations. Le déploiement du µ-outil sous forme de service web est lui aussi possible, nous ne détaillerons cependant pas cette alternative qui sort du contexte de cet article.

Une fois définis le cadre du collecticiel et le concept de µ-outil, il nous est maintenant possible de préciser l'élément informatique de base sur lequel il repose, à savoir l'agent logiciel et sa capacité à interagir (communiquer et coopérer) avec d'autres agents.

3 L'agent, support du µ-outil

Les agents logiciels sont largement utilisés dans la communauté CSCW. Les principales caractéristiques des agents (autonomie, adaptabilité, coopération et communication) permettent, d'une part, de gérer efficacement des composants distribués, hétérogènes et autonomes, et, d'autre part, de faciliter les échanges d'informations et le partage de ressources entre les composants [16], [9]. Même si l'agent peut être conçu avec une granularité moindre que le µ-outil, il n'en demeure pas moins vrai que les propriétés du premier correspondent à celles attendues par le second, surtout s'il s'agit de µ-outils coopératifs (*MOC*).

3.1 Eléments de modélisation

La définition des agents présentée dans cet article est adaptée du modèle de RASMUSSEN. Celui-ci distingue trois niveaux de comportements : le comportement réflexe, le comportement à base de règles et le comportement à base de connaissances, avec interprétation, décision et plan d'action. Nous l'avons interprété comme un modèle de processus des agents, dont les comportements sont adaptés aux tâches assignées par leur concepteur. Dans [10], nous avons proposé l'architecture générale de tels agents. Celle-ci respecte les trois propriétés d'indépendance, de communication et d'intelligence (Fig. 4). Elle est en outre inspirée de la théorie de modularité de l'esprit de Jerry FODOR (1981). L'architecture cognitive propre à l'agent intègre cinq modules dont la fonction est de gérer les connaissances, la perception, la communication, le contrôle et le raisonnement. Le modèle dynamique de ces modules internes est lui aussi présenté dans [10].

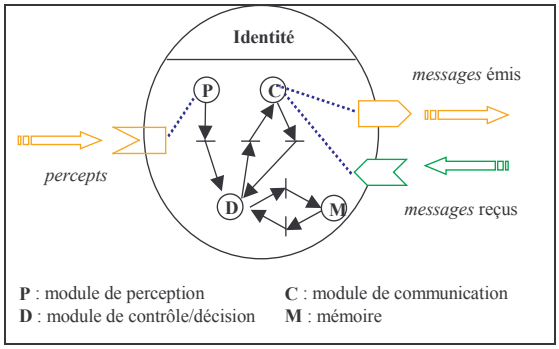

Figure 4. Architecture cognitive de l'agent

Ce qui vient d'être décrit ne concerne que l'environnement interne de l'agent. Réunis au sein d'un système, ceux-ci peuvent être hétérogènes, avoir des modalités d'interactions variées, des comportements différents, etc. Aussi doit-on compléter ce premier modèle individuel par un modèle des règles de composition entre agents, c'est-à-dire de leur(s) modalité(s) d'interaction.

3.2 Agents communicants et coopérants

La première modalité d'interaction entre un agent et sa communauté d'appartenance concerne la communication. Nous insistons sur le fait que pour qualifier un agent d'intelligent, avec toutes les précautions d'usage de ce qualificatif parfois galvaudé, il est important de prouver ses capacités à communiquer dans un but individuel (exemple : réaliser une tâche) ou collectif (exemple : s'associer à

d'autres agents pour réaliser une activité). Sans intentionnalité minimale, il n'y a donc pas d'intelligence agentive possible.

Les comportements individuels et coopératifs des agents sont variés : initialisations, planification des actions, émission et réception de documents et de messages, recherche de documents ou d'information, supervision de procédures. Chacun de ces services correspond à la mise en œuvre de compétences.

Pour décrire la communication entre agents (information ou dialogue pour la coopération), il est possible d'utiliser le langage KQML, dérivé de la théorie pragmatique des actes de langages. La forme générale d'un acte de langage est donnée par SEARLE sous l'expression *F(p)* où

F = {Affirmer, Demander, Promettre, Exprimer, Déclarer}

et *p* est une proposition. Pour notre part, le format que retenu pour un acte de langage est défini par le quintuplet :

A = <Intention, Emetteur, Récepteur, Langage, Message>.

Celui-ci permet de représenter le contexte, l'intention et le message de la communication de l'agent.

La compétence communicationnelle des agents est primordiale pour développer de futurs collecticiels et µ-outils plus flexibles et complexes que ceux actuellement proposés. L'apport d'agents intégrés dans des systèmes coopératifs concerne tout particulièrement :

- la prise en charge de tâches répétitives et la délégation de tâches sans intérêt pour le groupe d'utilisateurs (d'acteurs) ;
- la prise de décision par compréhension du contexte de l'activité (pertinence) ;
- la personnalisation de l'information (préférences, buts et capacités des utilisateurs / acteurs) ;
- l'interactivité naturelle (modalités et présentation) ;
- l'adéquation aux systèmes en réseau coopératifs.

Bien sûr, il s'agit là de perspectives offertes ; la plate-forme de recherche PLACID, que nous allons décrire, n'assurant ces fonctionnalités que partiellement à ce jour.

4 Intégrer les µ-outils : PLACID

Les activités coopératives considérées dans cet article sont celles de conception collaborative de produits, que ce soit en situation de conception distribuée ou de co-conception. Dans de telles activités, les pratiques collectives que l'on peut proposer d'instrumenter sont principalement : l'allocation des tâches selon les compétences des acteurs, la synchronisation des actions et la synchronisation cognitive pour partager les connaissances, la gestion des conflits, ainsi que de multiples actions de communication [4]. Les systèmes coopératifs à concevoir doivent donc offrir des fonctionnalités propres au déroulement de l'activité collective et permettre ainsi aux partenaires de la conception de coopérer pour :

- identifier les buts et partager leurs définitions,
- déterminer et distribuer les sous-buts associés,
- répartir les tâches à réaliser,
- suivre l'évolution de l'activité,
- évaluer les résultats de la conception collective.

4.1 Architecture de PLACID

PLACID a pour principale fonction de faciliter l'utilisation de µ-outils collaboratifs. Cette plate-forme logicielle de laboratoire a d'abord été dédiée à la co-conception, même si elle supporte un champ plus étendu d'applications. Dans ce premier cadre, elle offre des services pour l'utilisation d'un environnement de co-conception virtuelle (partage d'objets, services de gestion de tâches, services de communications et peut-être outils d'aide à la décision). Les différentes déclinaisons de PLACID permettent aussi bien l'usage ponctuel d'un µ-outil de conception que la constitution d'un véritable "bureau virtuel de co-conception" (*ie*, atelier).

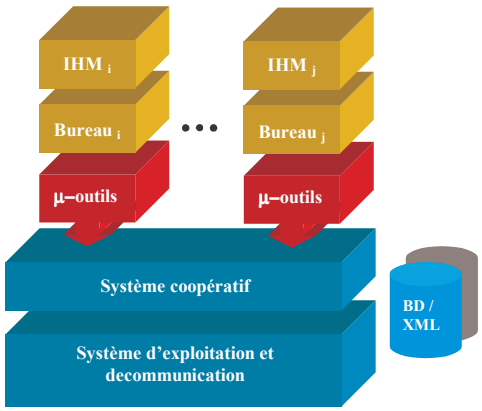

Figure 5. Architecture stratifiée et modulaire de PLACID

L'architecture de PLACID (Fig. 5) comprend cinq couches. Cette stratification permet une certaine modularité, donc des possibilités d'utilisation multi-plate-formes et d'évolution. Les fonctions des couches de PLACID sont les suivantes :

- couche 1, gestion de l'interface utilisateur (présentation, interactions et contrôle du dialogue) dans un contexte collaboratif, donc multi-utilisateurs ;
- couche 2, gestion de l'espace de travail (bureau de conception virtuel, par exemple) organisé en fonction du contexte d'usage (activité privée ou collaborative) ;
- couche 3, gestion des outils d'aide à la co-conception (µ-outils de conception, µ-outils de coopération et éventuellement autres outils de conception) ;
- couche 4, gestion de l'organisation des co-acteurs et des travaux collaboratifs (couche d'agents outils spécialisés dans la coordination de groupe, la gestion des objets partagés, etc.) ;
- couche 5 : système d'exploitation et gestion des communications de bas niveau.

4.2 Orientation agent de PLACID

La plate-forme PLACID a été développée selon le paradigme agent. Les agents de la plate-forme sont de type *application* (µ-outils d'aide à la co-conception), *coordinateur*/*médiateur*, *système* et *interface*. Le système d'agents assure, quant à lui, l'organisation et le contrôle de la communauté d'agents. L'utilisation effective du système (via une interface elle-même conçue selon une approche agent) se fait dans un contexte multi-utilisateurs.

PLACID permet ainsi de supporter un répertoire varié de configurations d'activité : de l'utilisation d'un µ-outil de conception unique à la constitution d'un véritable bureau virtuel de co-conception. Cette performance est garantie par la conception orientée agent de la plate-forme et le choix de CORBA (***C**ommon **O**bject **R**equest **B**roker **A**rchitecture*) pour la gestion des échanges et de l'information partagée. La couche médiane de PLACID, constituée des agents outils de coopération (Fig. 6), se décompose en deux niveaux :

- des agents *médiateurs*, proches des utilisateurs (acteurs) et des µ-outils ;
- des agents *outils* (exécutants, opérateurs) dotés des compétences indispensables à la coopération.

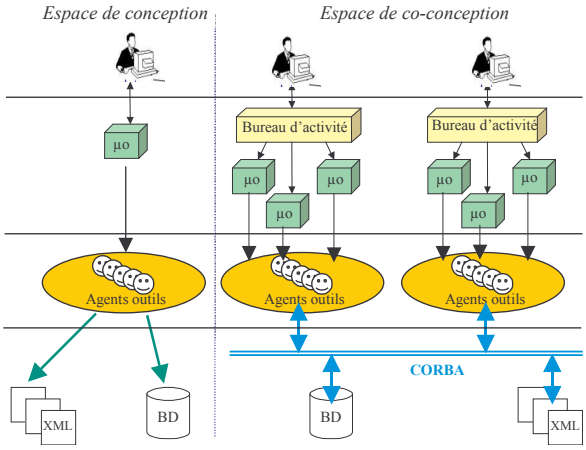

Figure 6. Architecture agent de PLACID

4.3 Intégration par agentification

Après avoir conçu la plate-forme agent PLACID, nous avons développé un processus d'agentification pour faciliter l'intégration des µ-outils coopératifs. Ce processus suit certaines propositions méthodologiques faites dans la définition du langage A_UML [20]. Celles-ci sont toutefois complétées par nos propres propositions méthodologiques, à savoir :

- réaliser le diagramme de cas d'utilisation des µ-outils (services attendus pour l'activité cible), et pour chacun des cas (usages) identifiés, réaliser les trois phases suivantes ;
- réaliser le diagramme de classes mettant en relation les agents (µ-outils) concernés par l'usage (on peut aussi faire usage du diagramme de collaboration) ;
- définir le comportement de chaque agent au moyen d'un diagramme d'états ou d'activités ;
- sur la base de scénarios d'usage, réaliser les diagrammes de séquence qui précisent les échanges de messages (et leur ordonnancement) entre les agents concernés par les scénarios.

Ces principes ont été suivis pour modéliser puis concevoir une application basique de réunion électronique, nommée « Papoticiel » (Fig. 7). Cet outil, outre son aspect communicationnel, met en relation des applications de coopération aussi variées que la gestion d'un groupe d'utilisateurs, la maintenance d'un agenda, la gestion d'une mémoire de groupe au travers de l'archivage des réunions et de ses éléments, ainsi que l'édition partagée de compte-rendu de réunion. Il nécessite le déploiement de 8 types d'agents : les *agents micro-outil,* des *agents utilisateur*, des *agents d'accueil (ou bureau)*, un *agent groupe*, un *agent communication*, un *agent d'archivage* et un *agent BD*.

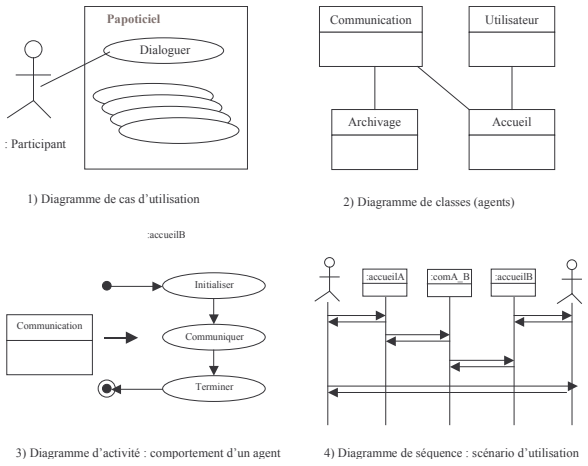

Figure 7. Méthode de conception agent : exemple du papoticiel.

Un exemple plus « consistant » de l'application du concept de µ-outil et de PLACID va être maintenant développé, il concerne le développement d'outils d'aide à l'analyse fonctionnelle collaborative.

5 Exemple : des µ-outils d'analyse fonctionnelle collaborative

Nous venons d'évoquer que pour valider le concept de µ-outils nous avions développé une application élémentaire, à savoir le « Papoticiel ». Depuis nous avons lancé trois chantiers de développement de µ-outils coopératifs : un premier pour l'aide au déploiement de la méthodologie TRIZ (*Teoria Reschenia Izobretateliskih Zadaci*, théorie de résolution des problèmes innovants) [31], un second pour l'évaluation des performances en ingénierie de systèmes manufacturiers [6] et un troisième pour l'activité d'analyse fonctionnelle.

L'analyse fonctionnelle (AF) est une méthode systématique et structurée de conception. Elle permet de décrire le système à concevoir (produit, process, logiciel,

service, etc.) sous forme de fonctions et ce, afin de répondre de façon satisfaisante aux besoins de l'utilisateur final. Ceci se traduit par la rédaction d'un cahier des charges fonctionnel. Compte tenu du nombre de variables manipulées, l'AF est souvent réalisée dans le cadre d'une activité collaborative. Celle-ci est encore souvent guidée et animée par un expert de la spécification fonctionnelle : les autres acteurs parties prenantes étant des ingénieurs d'études spécialisés dans tel ou tel domaine.

L'AF ayant la soixantaine, il existe de très nombreuses présentations de cette méthode [25]. Celles-ci proposent un plan d'action global, qui relève d'une description macroscopique de l'activité de spécification fonctionnelle. Notre démarche consiste à identifier puis à rendre modulaire, les tâches associées. Ceci nous a permis de définir un ensemble de douze µ-outils, puis de les développer conformément au processus *ICI* (Fig. 3). Dans cette section, après l'évocation du contexte organisationnel de l'activité d'AF, nous reprendrons chacune des trois phases de ce processus.

5.1 Organisation de l'activité d'AF

Les situations de travail coopératif pour lesquelles nous proposons une assistance à base de µ-outils, sont caractérisées par deux niveaux de flexibilité : 1) l'organisation matricielle des projets et des activités, basée sur l'association des compétences des acteurs participants à l'activité [21], 2) le fait que les activités de conception sont souvent organisées de façon opportuniste [28]. Pour le cas particulier de l'analyse fonctionnelle développé ici, cela se traduit, 1) par considérer le groupe de travail comme une communauté multi-métiers, et 2) par faciliter l'auto-organisation du groupe de travail, notamment par l'usage des MOC, conçus pour offrir au collectif l'espace nécessaire à la coopération (prescrite ou émergente), la négociation et la concertation.

5.2 Identification des µ-outils d'AF

Cette phase doit permettre à des concepteurs mécaniciens et des informaticiens d'identifier les µ-outils susceptibles d'apporter une assistance logicielle à l'analyse fonctionnelle. Ce processus d'identification conduit à l'élaboration du diagramme SADT, point de départ de la conception des µ-outils (12 dans ce cas) :

- analyse de l'activité et identification de graphes d'activité de référence (*cf* figure 9, *Milex→Beso→Devo*, *Isys→Caraf→Hiera ou Flux→Conta→Granu*).
- construction des actigrammes SADT de référence, à partir de l'analyse précédente (Fig. 8),
- identification des µ-outils potentiels à partir des actigrammes SADT.

Notons bien que notre idée, en procédant ainsi, n'est pas de figer un plan d'action en définissant un « *one best way* », donc un enchaînement rigide de tâches, mais d'apprécier la variété des graphes possibles de l'activité de référence. La démarche d'AF pourra alors s'appuyer sur l'utilisation des 12 µ-outils identifiés :

- les µ-outils d'AF externe, *Milex, Devo, Beso, Isys, Caraf et Hiera,* pour respectivement définir les milieux extérieurs (limites du système), définir l'évolution du système (cycle de vie), définir les besoins, établir la liste des fonctions (inventaire systémique), caractériser les fonctions, hiérarchiser les fonctions.
- les µ-outils d'AF interne, *Nomen, Flux, Conta, Granu, Fast, Coll,* pour respectivement créer une nomenclature, réaliser un diagramme de flux, réaliser un diagramme de contact, assurer la granularité adapté, réaliser un FAST et collecter les données du processus.

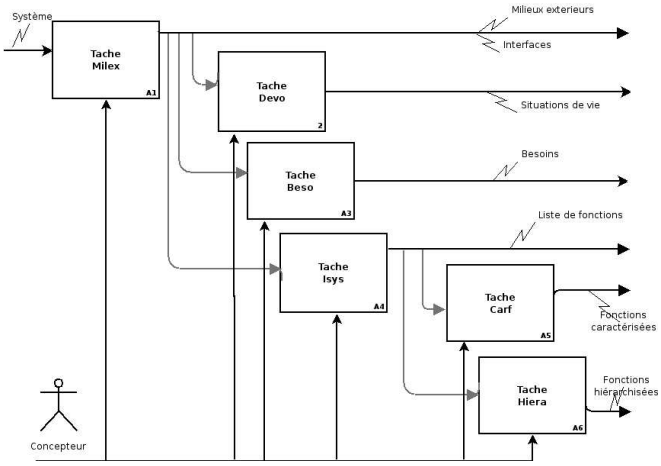

Figure 8. Actigramme SADT pour l'AF externe seule.

5.3 Conception des µ-outils d'AF

La deuxième phase de la méthode *ICI* consiste à spécifier, concevoir (diagrammes UML) et développer (pour nous, en Java) les µ-outils et leurs interfaces, en incluant le processus d'agentification présenté ci-dessus.

a) Spécification des µ-outils

Le diagramme de la figure 9 présente le contexte d'utilisation des µ-outils d'AF. Une activité d'analyse fonctionnelle (interne ou externe) peut être déclenchée à l'initiative d'un membre du groupe. Ce diagramme est facilement déduit des actigrammes SADT (Fig. 8) construits pour modéliser l'activité de référence d'AF.

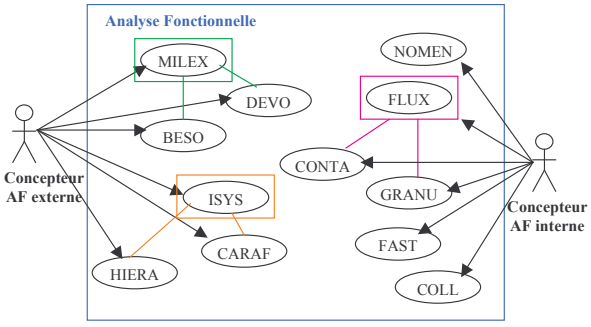

Figure 9. Cas d'utilisation des 12 µ-outils d'AF

La construction du diagramme de classe d'un µ-outil est standardisée (Fig. 10) : une classe pour le µ-outil lui-même, une classe pour son IHM et un ensemble de classes pour gérer les objets de conceptions traités par le µ-outil (*ME* et *Interface* pour le µ-outil *Milex*, par exemple). Un µ-outil supplémentaire permet de créer des processus tels que ceux identifiés sur la figure 9 (*Milex→Beso→Devo* par exemple). L'utilisation des µ-outils peut alors être prescrite par une méthodologie spécifique.

Dernier type de diagramme réalisé lors de la phase de conception, le diagramme de collaboration. Celui-ci permet de décrire la communication et l'ordonnancement des agents impliqués dans un scénario de référence.

b) Conception des agents

En complément des 8 agents µ-outils déjà disponibles sur PLACID (*Coordinateur, Informateur, Processus, Projet, Papoticiel, Vote, Agenda, Idée*), l'outillage de l'activité d'AF nécessite le déploiement de 3 autres types d'agents (Fig. 11) : 1) les 12 agents µ-outils (*Milex, Devo, Beso, Isys, Hiera, Caraf, Nomen, Flux, Conta, Granu, Fast, Coll*), 2) un agent *bureau* pour assister la session de travail du concepteur, 3) un agent *clientCorba*, un agent *serveurCorba* et un agent *BD/XML* pour gérer la coopération, la mémorisation et le contexte multi-utilisateur. Le tableau 2 recense les différentes compétences mises en œuvre par chaque agent.

c) Conception des interfaces des µ-outils

Pour ce qui concerne le développement des interfaces des µ-outils, parmi les options possibles [5], une démarche incrémentale a été retenue. Sa performance est double :

- du point de vue de la modélisation des interfaces, elle permet une analyse fine des interactions (acteurs / système d'aide à l'activité collaborative). Pour ce faire, une méthode couplant SADT et les RdP a été retenue [1]. L'intérêt majeur de cette association est d'effectuer en parallèle : a) la structuration fonctionnelle du système en termes de tâches (par SADT), et b) la description de son comportement dynamique (par le RdP associé) ;
- du point de vue de l'architecture logicielle, elle permet de suivre les standards du modèle *PAC* [3], bien adaptés aux systèmes interactifs et aux collecticiels.

Agents	Compétences
Agents AF externe *Agents AF interne*	• Gestion des µ-outils : interactions utilisateurs et IHM associées. • Réalisation des tâches d'AF externe : *Milex, Beso, Devo, Isys, Hiera, Caraf,* • Réalisation des tâches d'AF interne : *Nomen, Flux, Conta, Granu, Fast, Coll*
Bureau	• Communication avec *ClientCorba.* • Coordination des agents des µ-outils. • Transmission des données de projets. • Communication à l'utilisateur (acteur) des informations de coopération.
Client CORBA	• Gestion des communications (messages et événements) côté client. • Communication avec l'agent *ServeurCorba* (requêtes de l'utilisateur). • Transmission des informations coopératives au bureau de l'utilisateur.
Serveur CORBA	• Gestion des communications (messages et événements) côté serveur. • Communication avec l'agent *ClientCorba* (réponses à l'utilisateur). • Gestion des accès concurrents aux données de projets. • Gestion des sessions des utilisateurs. • Envoi de messages coopératifs.
BD et/ou *XML*	• Gestion de fichiers XML (création, ouverture, enregistrement, fermeture, destruction). • Gestion des échanges (requêtes/réponses) avec une BD. • Transfert de données.

Tableau 2. Compétences des agents d'AF collaborative

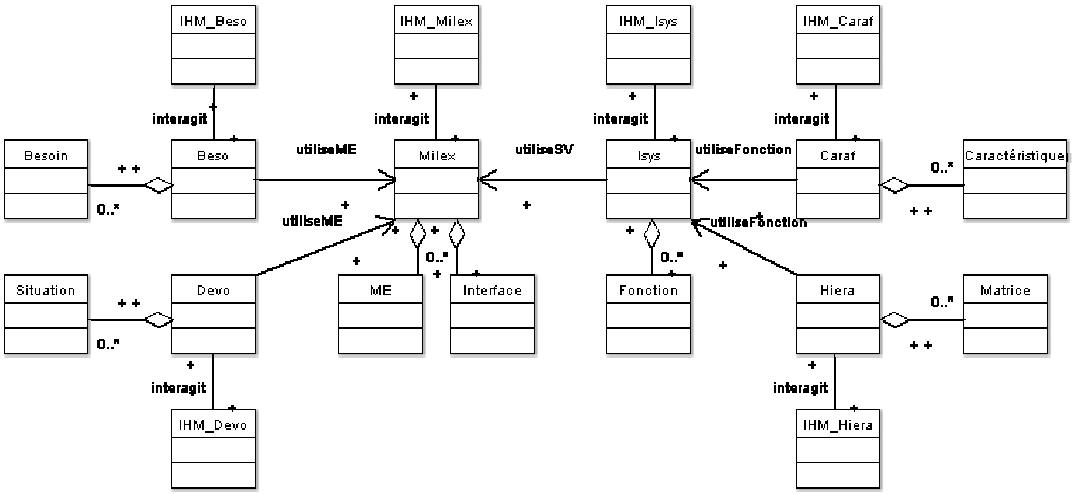

Figure 10. Diagramme de classe pour les µ-outils d'AF externe seule

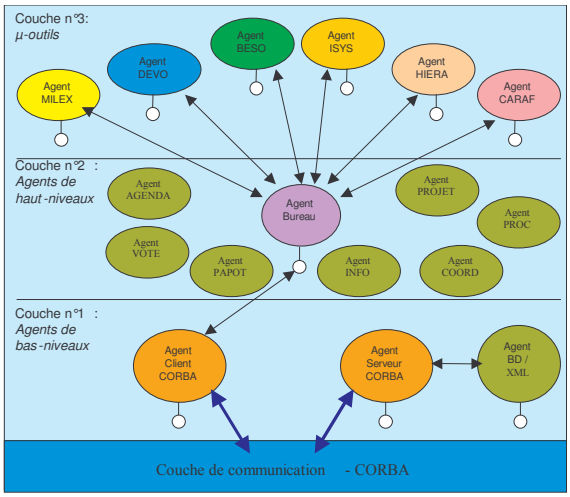

Figure 11. Structuration des agents d'AF externe sous la plate-forme PLACID

En guise d'illustration, la figure 12 présente une vue d'écran du bureau d'AF du concepteur «fougeres», lors de la réalisation de la tâche coopérative « Beso » (définition des besoins), pour l'analyse d'un pied d'appareil photo.

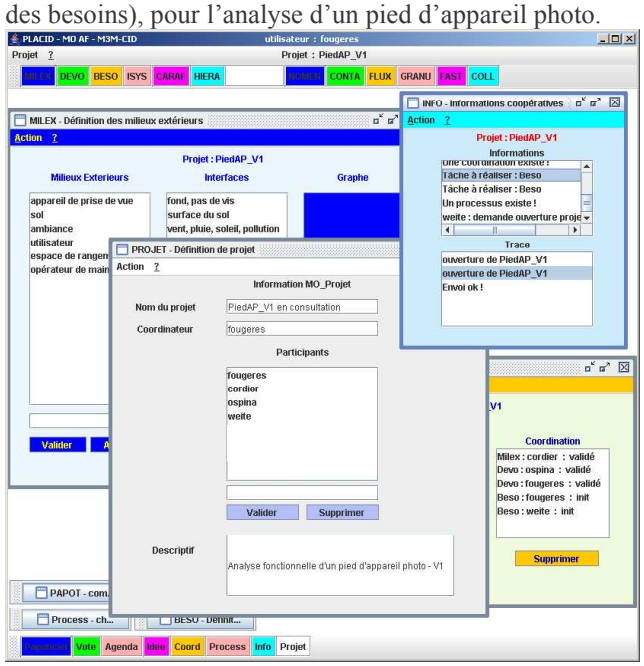

Figure 12. Bureau de PLACID :
Exemple d'AF collaborative d'un pied d'appareil photo

Sept µ-outils sont ouverts dans son bureau d'activité :

- *Milex et Beso*, pour sa tâche de définition de besoins ;
- *PROJET,* qui lui fournit les informations de contexte de réalisation du projet ;
- *COORD,* qui lui permet de définir puis de suivre l'évolution de l'activité en fonctions des tâches distribuées aux différents concepteurs ; dans le cas présent on constate que deux concepteurs ont pour rôle de définir les besoins ;
- *Process,* qui lui permet de contrôler et de suivre un processus qui définit l'ordre des tâches à mener pour l'AF ;

- *INFO,* qui lui permet de visualiser sa propre activité et celles des autres. Il peut alors communiquer pour mieux coordonner sa production ; dans le cas présent le µ-outil l'informe que le concepteur « weité » ouvre le même projet pour lui aussi travailler sur la définition des besoins. Un échange de messages géré par le µ-outil *PAPOT* permet aux deux concepteurs de s'accorder.

5.4 L'intégration des µ-outils dans PLACID

L'intégration finale des µ-outils d'AF dans PLACID est réalisée en deux étapes. La première correspond à l'agentification des douze µ-outils conçus pour l'AF collaborative. La seconde consiste à définir, puis à inclure, un agent *ClientCorba* à PLACID, qui sera référencé par chaque bureau d'activité d'utilisateurs. La figure 13 présente le cas particulier du µ-outil *Milex*. Comme le montre les cercles portés sur la figure, celui-ci, en plus de son association avec la classe *IHM_Milex* et de sa composition de classes *ME* et *Interface*, hérite bien évidemment de la classe *Agent MO* (Micro-Outils).

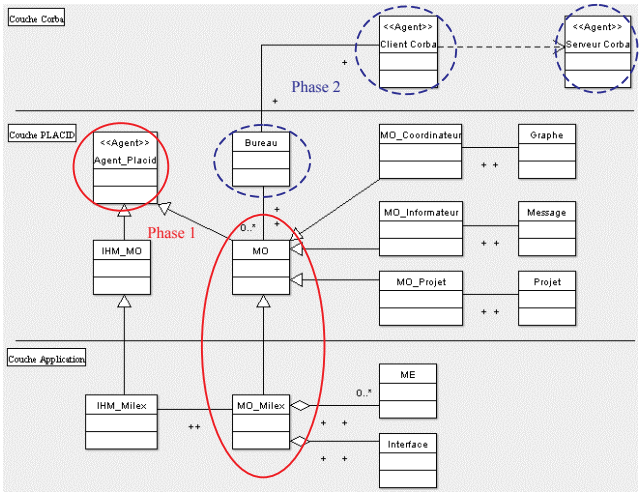

Figure 13. Intégration du µ-outil dans PLACID :
Exemple du µ-outil MILEX

Une fois cette étape terminée, commence alors les multiples cycles de validation de l'atelier d'AF. Celui-ci n'ayant qu'une vocation de recherche, une validation en grandeur industrielle n'a pas encore été réalisée.

6 Conclusion

Cet article a présenté le concept original de micro-outil logiciel et la plate-forme agent PLACID (*Plate-forme Logicielle d'Aide à la Conception Innovante et Distribuée)*. La fonction initiale de PLACID est d'assister la co-conception distribuée et plus généralement, les activités collaboratives. Cette plate-forme de recherche, dont le développement est inscrit dans le domaine technologique des collecticiels, présente l'originalité d'intégrer des micro-outils et de les supporter grâce au paradigme agent. Celui-

ci concerne aussi bien la modélisation et le développement des différentes couches de PLACID que celles de ses interfaces. Un cas d'école a servi à valider la plate-forme, à savoir celui de l'analyse fonctionnelle collaborative. Sans détailler outre mesure ce point, nous avons évoqué que deux nouveaux chantiers de µ–outils respectant le processus *ICI* ont été lancés : un premier pour l'aide au déploiement de la méthodologie TRIZ (théorie de résolution des problèmes innovants), un second pour l'évaluation des performances en ingénierie de systèmes manufacturiers.

Pour la poursuite de nos travaux, une perspective intéressante se dégage avec la généralisation des concepts du µ-outil logiciel à d'autres activités collaboratives, distinctes selon leurs objets, leurs acteurs concrets, mais similaires selon les modalités et les outils d'assistance. Nul doute qu'une telle perspective ouvre des directions dans les domaines du génie logiciel et de l'ingénierie des connaissances, étant donnée leur dimension collaborative.

Références

[1] M. Abed, H. Ezzedine et C. Kolski C. Modélisation des tâches dans la conception et l'évaluation des systèmes interactifs : la méthode SADT/Petri. *Analyse et conception de l'IHM*, sous la direction de C. Kolski, pp. 145-174, Paris : Hermès, 2001.

[2] J.-C. Courbon et S. Tajan. *Groupware et intranet. Vers le partage des connaissances*, 2ème édition, Paris : DUNOD, 1999.

[3] J. Coutaz et L. Nigay. Architecture logicielle conceptuelle des systèmes interactifs. *Analyse et conception de l'IHM*, sous la direction de C. Kolski, pp. 207-246, Paris : Hermès, 2001.

[4] F. Darse et P. Falzon. La conception collective : une approche de l'ergonomie cognitive. *Coopération et conception*, sous la direction de G. de Terssac et E. Friedberg, pp. 123-135, Toulouse : Editions Octares, 1996.

[5] B. David. IHM pour les collecticiels. *Réseaux et systèmes répartis*, vol. 13, pp. 169-206, Paris : Hermès, nov. 2001.

[6] I. Deniaud, J.-P. Micaëlli et A.-J. Fougères. Déployer et définir de façon collaborative une performance en ingénierie système manufacturière : l'exemple de la réactivité. *Actes du 6ème Congrès international de Génie Industriel*, Besançon, 7-10 juin 2005.

[7] C.A. Ellis, S.J. Gibbs et G.L. Rein. Groupware : some issues and experiences. *Communications of ACM* 34(1), p. 38-58, 1991.

[8] C.A. Ellis et J. Wainer. A conceptual model of Groupware. In *Proceedings of CSCW'94*, ACM Press, pp. 79-88, 1994.

[9] J. Ferber. *Les systèmes multi-agents. Vers une intelligence collective*, Paris : InterEditions, 1995.

[10] A.-J. Fougères. Des agents communicants pour simuler et détecter des épidémies. *Revue Ingénierie des Systèmes d'Information*, (8)1, pp. 91-112, Paris : Hermès, 2003.

[11] A.-J. Fougères. Agents to cooperate in distributed design. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetic* (SMC'04), pp. 2629-2634, The Hague, Netherlands, October 10-13 2004.

[12] A.-J. Fougères. Agent-based micro-tools development for a co-operative design platform. *Proceedings of the ITI 3rd International Conference on Information & Communications Technology* (*ICICT'05)*, Cairo, December 2005.

[13] O. Garro. Conception distribuée – de l'industrie… à l'industrie. *Actes des 4e Journées Francophones d'Intelligence Artificielle et Systèmes Multi-Agents* (JFIADSMA'96), Hermès, Port Camargue, 1-3 avril 1996.

[14] D. Hérin, B. Espinasse, E. Andonoff et C. Hanachi. Des systèmes d'information coopératifs aux agents informationnels. *Ingénierie des systèmes d'information*, sous la direction de Corine Cauvet et Camille Rosenthal-Sabroux, p. 209-244, Paris : Hermès, 2001.

[15] F. Hoogstoel. Une approche organisationnelle du travail coopératif assisté par ordinateur. Application au projet CO-LEARN, Thèse de Doctorat de l'Université des Sciences et Techniques de Lille, 1995.

[16] K. Kuutti. Activity Theory as a Potential Framework for Human-Computer Interaction Research. *Context and Consciousness. Activity Theory and Human-Computer Interaction*, Edited by Bonnie A. Nardi, Cambridge : MIT Press, pp. 17-44, 1996.

[17] J. Lonchamp. *Le Travail Coopératif et ses technologies*. Paris : Hermès, 2003.

[18] B. Maggi et V. Lagrange. *Le travail collectif dans l'industrie à risque. Six points de vue de chercheurs étayés et discutés*. Toulouse : Octares Editions, 2002.

[19] J.-P. Micaëlli et J. Forest. *Artificialisme : introduction à une théorie de la conception*. Lausanne : Presses Polytechniques et Universitaires Romandes, 2003.

[20] J. Odell, H.V.D. Parunak et B. Bauer B. Extending UML for agents. *Proceedings of Agent-Oriented Information Systems*, *Workshop at the 17th National conference on Artificial Intelligence*, Austin, Texas, july, 30, 2000.

[21] J. Perrin, M.-C. Villeval et Y. Lecler. Les différents modes de coordination mobilisés pour promouvoir la coopération dans la démarche de concurrent engineering – Trois études de cas en Rhône-Alpes. *Coopération et conception*, sous la direction de G. de Terssac et E. Friedberg, Toulouse : Editions Octares, 1996.

[22] J.C. Routier et P. Mathieu. Une contribution du multi-agent aux applications de travail coopératif. *Technique et science informatiques*, 14(4), pp. 473-500, Paris : Hermès, 1995.

[23] K. Schmidt et L. Bannon. Taking CSCW seriously. *Computer Supported Cooperative Work Journal*, 1(1), 1992.

[24] Y. Shoham. Agent-Oriented Programming. *Artificial Intelligence*, vol. 60, pp. 51-92, 1993.

[25] R. Tassinari. *Pratique de l'Analyse Fonctionnelle*. 2e édition, Paris : Dunod, 1997.

[26] G. Terssac et E. de Friedberg. *Coopération et conception*. sous la direction de Gilbert de Terssac et Erhard Friedberg, Collection Travail, Toulouse : Editions Octares, 1996.

[27] E. Van Handenhoven et P. Trassaert. Design knowledge and design skills. *Proceedings of the International Conference on Engineering Design* (ICED 99), Munich, Allemagne, 24-26 août, 1999.

[28] W. Visser. Conception individuelle et collective. Approche de l'ergonomie cognitive. Rapport de recherche de l'INRIA, n°4257, septembre 2001.

[29] L.S. Vygotsky. *Mind and Society*. Cambridge MA: Harvard University Press, 1978.

[30] G. Wagner. Towards Agent-Oriented Information Systems. Technical Report, University of Leipzig, Germany, March 1999.

[31] P.-A. Weité, A.-J. Fougères et C. Gazo. Les micro-outils, vecteur d'appropriation des nouvelles méthodologies de conception et d'innovation. *Actes du 6ème Congrès international de Génie Industriel*, Besançon, 7-10 juin 2005.